Free Questions for Professional-Machine-Learning-Engineer by actualtestdumps

Shared by Rice on 29-01-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: **MultipleChoice**

You are creating a social media app where pet owners can post images of their pets. You have one million user uploaded images with hashtags. You want to build a comprehensive system that recommends images to users that are similar in appearance to their own uploaded images.

What should you do?

Options:

A- Download a pretrained convolutional neural network, and fine-tune the model to predict hashtags based on the input images. Use the predicted hashtags to make recommendations.

B- Retrieve image labels and dominant colors from the input images using the Vision API. Use these properties and the hashtags to make recommendations.

C- Use the provided hashtags to create a collaborative filtering algorithm to make recommendations.

D- Download a pretrained convolutional neural network, and use the model to generate embeddings of the input images. Measure similarity between embeddings to make recommendations.

Answer:

D

Explanation:

The best option to build a comprehensive system that recommends images to users that are similar in appearance to their own uploaded images is to download a pretrained convolutional neural network (CNN), and use the model to generate embeddings of the input images. Embeddings are low-dimensional representations of high-dimensional data that capture the essential features and semantics of the data. By using a pretrained CNN, you can leverage the knowledge learned from large-scale image datasets, such as ImageNet, and apply it to your own domain. A pretrained CNN can be used as a feature extractor, where the output of the last hidden layer (or any intermediate layer) is taken as the embedding vector for the input image. You can then measure the similarity between embeddings using a distance metric, such as cosine similarity or Euclidean distance, and recommend images that have the highest similarity scores to the user's uploaded image. Option A is incorrect because downloading a pretrained CNN and fine-tuning the model to predict hashtags based on the input images may not capture the visual similarity of the images, as hashtags may not reflect the appearance of the images accurately. For example, two images of different breeds of dogs may have the same hashtag #dog, but they may not look similar to each other. Moreover, fine-tuning the model may require additional data and computational resources, and it may not generalize well to new images that have different or missing hashtags. Option B is incorrect because retrieving image labels and dominant colors from the input images using the Vision API may not capture the visual similarity of the images, as labels and colors may not reflect the fine-grained details of the images. For example, two images of the same breed of dog may have different labels and colors depending on the background, lighting, and angle of the image. Moreover, using the Vision API may incur additional costs and latency, and it may not be able to handle custom or domain-specific labels. Option C is incorrect because using the provided hashtags to create a collaborative filtering algorithm may not capture the visual similarity of the images, as collaborative filtering relies on the ratings or preferences of users, not the features of the images. For example, two images of different animals may have similar ratings or preferences from users, but they may not look similar to each other. Moreover, collaborative filtering may suffer from the cold start problem, where new images or users that have no ratings or preferences cannot be recommended.Reference:

Image similarity search with TensorFlow

Image embeddings documentation

Pretrained models documentation

Similarity metrics documentation

Question 2

Question Type: MultipleChoice

You developed a BigQuery ML linear regressor model by using a training dataset stored in a BigQuery table. New data is added to the table every minute. You are using Cloud Scheduler and Vertex AI Pipelines to automate hourly model training, and use the model for direct inference. The feature preprocessing logic includes quantile bucketization and MinMax scaling on data received in the last hour. You want to minimize storage and computational overhead. What should you do?

Options:

A- Create a component in the Vertex AI Pipelines directed acyclic graph (DAG) to calculate the required statistics, and pass the statistics on to subsequent components.

B- Preprocess and stage the data in BigQuery prior to feeding it to the model during training and inference.

C- Create SQL queries to calculate and store the required statistics in separate BigQuery tables that are referenced in the CREATE MODEL statement.

D- Use the TRANSFORM clause in the CREATE MODEL statement in the SQL query to calculate the required statistics.

Answer:

D

Explanation:

The best option to minimize storage and computational overhead is to use the TRANSFORM clause in the CREATE MODEL statement in the SQL query to calculate the required statistics. The TRANSFORM clause allows you to specify feature preprocessing logic that applies to both training and prediction. The preprocessing logic is executed in the same query as the model creation, which avoids the need to create and store intermediate tables. The TRANSFORM clause also supports quantile bucketization and MinMax scaling, which are the preprocessing steps required for this scenario. Option A is incorrect because creating a component in the Vertex AI Pipelines DAG to calculate the required statistics may increase the computational overhead, as the component needs to run separately from the model creation. Moreover, the component needs to pass the statistics to subsequent components, which may increase the storage overhead. Option B is incorrect because preprocessing and staging the data in BigQuery prior to feeding it to the model may also increase the storage and computational overhead, as you need to create and maintain additional tables for the preprocessed data. Moreover, you need to ensure that the preprocessing logic is consistent for both training and inference. Option C is incorrect because creating SQL queries to calculate and store the required statistics in separate BigQuery tables may also increase the storage and computational overhead, as you need to create and maintain additional tables for the statistics. Moreover, you need to ensure that the statistics are updated regularly to reflect the new data.Reference:

BigQuery ML documentation

Using the TRANSFORM clause

Feature preprocessing with BigQuery ML

Question 3

Question Type: **MultipleChoice**

While running a model training pipeline on Vertex AI, you discover that the evaluation step is failing because of an out-of-memory error. You are currently using TensorFlow Model Analysis (TFMA) with a standard Evaluator TensorFlow Extended (TFX) pipeline component for the evaluation step. You want to stabilize the pipeline without downgrading the evaluation quality while minimizing infrastructure overhead. What should you do?

Options:

A- Add tfma.MetricsSpec () to limit the number of metrics in the evaluation step.

B- Migrate your pipeline to Kubeflow hosted on Google Kubernetes Engine, and specify the appropriate node parameters for the evaluation step.

C- Include the flag -runner=DataflowRunner in beam_pipeline_args to run the evaluation step on Dataflow.

D- Move the evaluation step out of your pipeline and run it on custom Compute Engine VMs with sufficient memory.

Answer:

C

Explanation:

The best option to stabilize the pipeline without downgrading the evaluation quality while minimizing infrastructure overhead is to use Dataflow as the runner for the evaluation step. Dataflow is a fully managed service for executing Apache Beam pipelines that can scale up and down according to the workload. Dataflow can handle large-scale, distributed data processing tasks such as model evaluation, and it can also integrate with Vertex AI Pipelines and TensorFlow Extended (TFX). By using the flag-runner=DataflowRunnerinbeam_pipeline_args, you can instruct the Evaluator component to run the evaluation step on Dataflow, instead of using the default DirectRunner, which runs locally and may cause out-of-memory errors. Option A is incorrect because addingtfma.MetricsSpec()to limit the number of metrics in the evaluation step may downgrade the evaluation quality, as some important metrics may be omitted. Moreover, reducing the number of metrics may not solve the out-of-memory error, as the evaluation step may still consume a lot of memory depending on the size and complexity of the data and the model. Option B is incorrect because migrating the pipeline to Kubeflow hosted on Google Kubernetes Engine (GKE) may increase the infrastructure overhead, as you need to provision, manage, and monitor the GKE cluster yourself. Moreover, you need to specify the appropriate node parameters for the evaluation step, which may require trial and error to find the optimal configuration. Option D is incorrect because moving the evaluation step out of the pipeline and running it on custom Compute Engine VMs may also increase the infrastructure overhead, as you need to create, configure, and delete the VMs yourself. Moreover, you need to ensure that the VMs have sufficient memory for the evaluation step, which may require trial and error to find the optimal machine type.Reference:

Dataflow documentation

Using DataflowRunner

Evaluator component documentation

Configuring the Evaluator component

Question 4

Question Type: **MultipleChoice**

You work at a gaming startup that has several terabytes of structured data in Cloud Storage. This data includes gameplay time data, user metadata, and game metadat

a. You want to build a model that recommends new games to users that requires the least amount of coding. What should you do?

Options:

A- Load the data in BigQuery. Use BigQuery ML to train an Autoencoder model.

B- Load the data in BigQuery. Use BigQuery ML to train a matrix factorization model.

C- Read data to a Vertex AI Workbench notebook. Use TensorFlow to train a two-tower model.

D- Read data to a Vertex AI Workbench notebook. Use TensorFlow to train a matrix factorization model.

Answer:

B

Explanation:

The best option to build a game recommendation model with the least amount of coding is to use BigQuery ML, which allows you to create and execute machine learning models using standard SQL queries. BigQuery ML supports several types of models, including matrix factorization, which is a common technique for collaborative filtering-based recommendation systems. Matrix factorization models learn latent factors for users and items from the observed ratings, and then use them to predict the ratings for new user-item pairs. BigQuery ML provides a built-in function calledML.RECOMMENDthat can generate recommendations for a given user based on a trained matrix factorization model. To use BigQuery ML, you need to load the data in BigQuery, which is a serverless, scalable, and cost-effective data warehouse. You can use thebqcommand-line tool, the BigQuery API, or the Cloud Console to load data from Cloud Storage to BigQuery. Alternatively, you can use federated queries to query data directly from Cloud Storage without loading it to BigQuery, but this may incur additional costs and performance overhead. Option A is incorrect because BigQuery ML does not support Autoencoder models, which are a type of neural network that can learn compressed representations of the input data. Autoencoder models are not suitable for recommendation systems, as they do not capture the interactions between users and items. Option C is incorrect because using TensorFlow to train a two-tower model requires more coding than using BigQuery ML. A two-tower model is a type of neural network that learns embeddings for users and items separately, and then combines them with a dot product or a cosine similarity to compute the rating. TensorFlow is a low-level framework that requires you to define the model architecture, the loss function, the optimizer, the training loop, and the evaluation metrics. Moreover, you need to read the data from Cloud Storage to a Vertex AI

Workbench notebook, which is an instance of JupyterLab that runs on a Google Cloud virtual machine. This may involve additional steps such as authentication, authorization, and data preprocessing. Option D is incorrect because using TensorFlow to train a matrix factorization model also requires more coding than using BigQuery ML. Although TensorFlow provides some high-level APIs such as Keras and TensorFlow Recommenders that can simplify the model development, you still need to handle the data loading and the model training and evaluation yourself. Furthermore, you need to read the data from Cloud Storage to a Vertex AI Workbench notebook, which may incur additional complexity and costs.Reference:

BigQuery ML documentation

Using matrix factorization with BigQuery ML

Recommendations AI documentation

Loading data into BigQuery

Querying data in Cloud Storage from BigQuery

Vertex AI Workbench documentation

TensorFlow documentation

TensorFlow Recommenders documentation

Question 5

Question Type: **MultipleChoice**

You have recently developed a new ML model in a Jupyter notebook. You want to establish a reliable and repeatable model training process that tracks the versions and lineage of your model artifacts. You plan to retrain your model weekly. How should you operationalize your training process?

Options:

A- 1. Create an instance of the CustomTrainingJob class with the Vertex AI SDK to train your model.
2. Using the Notebooks API, create a scheduled execution to run the training code weekly.

B- 1. Create an instance of the CustomJob class with the Vertex AI SDK to train your model.
2. Use the Metadata API to register your model as a model artifact.
3. Using the Notebooks API, create a scheduled execution to run the training code weekly.

C- 1. Create a managed pipeline in Vertex Al Pipelines to train your model by using a Vertex Al CustomTrainingJoOp component.
2. Use the ModelUploadOp component to upload your model to Vertex Al Model Registry.
3. Use Cloud Scheduler and Cloud Functions to run the Vertex Al pipeline weekly.

D- 1. Create a managed pipeline in Vertex Al Pipelines to train your model using a Vertex Al HyperParameterTuningJobRunOp component.
2. Use the ModelUploadOp component to upload your model to Vertex Al Model Registry.
3. Use Cloud Scheduler and Cloud Functions to run the Vertex Al pipeline weekly.

Answer:

C

Explanation:

The best way to operationalize your training process is to use Vertex AI Pipelines, which allows you to create and run scalable, portable, and reproducible workflows for your ML models. Vertex AI Pipelines also integrates with Vertex AI Metadata, which tracks the provenance, lineage, and artifacts of your ML models. By using a Vertex AI CustomTrainingJobOp component, you can train your model using the same code as in your Jupyter notebook. By using a ModelUploadOp component, you can upload your trained model to Vertex AI Model Registry, which manages the versions and endpoints of your models. By using Cloud Scheduler and Cloud Functions, you can trigger your Vertex AI pipeline to run weekly, according to your plan.Reference:

Vertex AI Pipelines documentation

Vertex AI Metadata documentation

Vertex AI CustomTrainingJobOp documentation

ModelUploadOp documentation

Cloud Scheduler documentation

[Cloud Functions documentation]

Question 6

Question Type: **MultipleChoice**

Your work for a textile manufacturing company. Your company has hundreds of machines and each machine has many sensors. Your team used the sensory data to build hundreds of ML models that detect machine anomalies Models are retrained daily and you need to deploy these models in a cost-effective way. The models must operate 24/7 without downtime and make sub millisecond predictions. What should you do?

Options:

A- Deploy a Dataflow batch pipeline and a Vertex AI Prediction endpoint.

B- Deploy a Dataflow batch pipeline with the Runlnference API. and use model refresh.

C- Deploy a Dataflow streaming pipeline and a Vertex AI Prediction endpoint with autoscaling.

D- Deploy a Dataflow streaming pipeline with the Runlnference API and use automatic model refresh.

Answer:

D

Explanation:

A Dataflow streaming pipeline is a cost-effective way to process large volumes of real-time data from sensors. The RunInference API is a Dataflow transform that allows you to run online predictions on your streaming data using your ML models. By using the RunInference API, you can avoid the latency and cost of using a separate prediction service. The automatic model refresh feature enables you to update your models in the pipeline without redeploying the pipeline. This way, you can ensure that your models are always up-to-date and accurate. By deploying a Dataflow streaming pipeline with the RunInference API and using automatic model refresh, you can achieve sub-millisecond predictions, 24/7 availability, and low operational overhead for your ML models.Reference:

Dataflow documentation

RunInference API documentation

Automatic model refresh documentation

Preparing for Google Cloud Certification: Machine Learning Engineer Professional Certificate